Simulate Timer Job Solution for SharePoint 2013/Online using App Model & CSOM

Shariq-MSFT 9 Dec 2013 11:06 AM | 18

PLEASE NOTE: THIS IS STRICTLY A CONCEPT THAT I HAVE UTILIZED AND MAY/MAY NOT CONFIRM TO OFFICIAL MICROSOFT PRODUCT GROUP STATEMENT

With SharePoint 2013, the emphasis is on loosely coupled architecture that utilizes CSOM/REST. Going forward, it is recommended that custom Solutions like Web Parts be built as Apps or App parts, Event Receivers be Remote Event receivers etc. App scoped External Content Type for BCS is possible as well, and same goes for Workflows. In fact the Workflow framework utilizes Workflow Manager 1.0 that runs outside the SharePoint process.

So, the benefits are two-fold:

1) Secure SharePoint Farm from us 'Developers' writing against w3wp.exe :)

2) Ensure our solutions work seamlessly in SharePoint Online as well.

All good; but what happens to **Custom Timer Job** that we all have been writing in the past. Though I can still write it using Server Side Object Model (SSOM) for On Premises, it doesn't really confirm to the recommendation and what happens with SharePoint Online (SP-O)? No SSOM possible :(

Well, here is what I did to create a Custom Timer Job Simulator service that's uses App Model & CSOM. Now, this service doesn't inherits SPJobDefinition but instead this Windows Service runs based on a schedule and utilizes CSOM to perform what is required.

It's a two step process.

- Create a Provider Hosted App that uses App Only policy for SharePoint 2013/Online
- Create a Windows service that utilizes the code created above

Step 1. Create a Provider Hosted App that uses App Only policy

Steve Peschka has this nice blog post that highlights the use of Apps with App Only policy at Security in SharePoint Apps - Part 6. Also, Kirk Evans[MSFT] has a fantastic post on the subject at SharePoint 2013 App Only Policy Made

Easy.

This gives us a head start onto creating Provider Hosted Apps that utilizes 'AppOnly; policy. If you are working with On Premise then you have the flexibility to use either High Trust or Low Trust Apps. But in SP-O you can only use Low Trust. Setting up a low trust App for SP-O is simple with no additional configuration as o365 has an ACS. But for On-Premise please visit here to set it up.

1. For simplicity let's just use Low Trust App since it works with On-Premise & SP-O. So go ahead and create a SharePoint Provider Hosted App using Visual Studio 2012 (don't change the default option of using ACS).

Below is the snapshot of the code used inside my Low Trust App. It creates an 'AppOnlyAccessToken' that's used to create the 'ClientContext' which eventually updates a custom List. This could very well do any other task as well (things which are possible with the powerful CSOM). Verify that the App works fine with both On Premise & Online. Full sample code solution is attached below.

```
protected void Page_Load(object sender, EventArgs e)
    {
        //constant string SharePoint principal
        string SharePointPrincipal = "00000003-0000-0ff1-ce00-000000000000";

        var contextToken = TokenHelper.GetContextTokenFromRequest(Page.Request);

        Uri hostWeb = new Uri(Page.Request["SPHostUrl"]);

        string realm = TokenHelper.GetRealmFromTargetUrl(hostWeb);

        string appOnlyAccessToken = TokenHelper.GetAppOnlyAccessToken(SharePointPrincipal,
hostWeb.Authority, realm).AccessToken;

        using (ClientContext clientContext =
TokenHelper.GetClientContextWithAccessToken(hostWeb.ToString(), appOnlyAccessToken))
        {

            if (clientContext != null)
            {
                //ShariqTest is a custom List in my SharePoint site -both Om Prem & Online
                var myList = clientContext.Web.Lists.GetByTitle("ShariqTest");
                ListItemCreationInformation listItemCreate = new ListItemCreationInformation();
                Microsoft.SharePoint.Client.ListItem newItem = myList.AddItem(listItemCreate);
                newItem["Title"] = "Testing " ;
```

```
                newItem.Update();
                clientContext.ExecuteQuery();
            }
        }
    }
```

2. If you are looking at using only the High Trust App, then the process pretty much remains same. The code to get 'AppOnlyAccessToken' is a bit different. Please follow this and the attached sample that Steve Peschka shared.

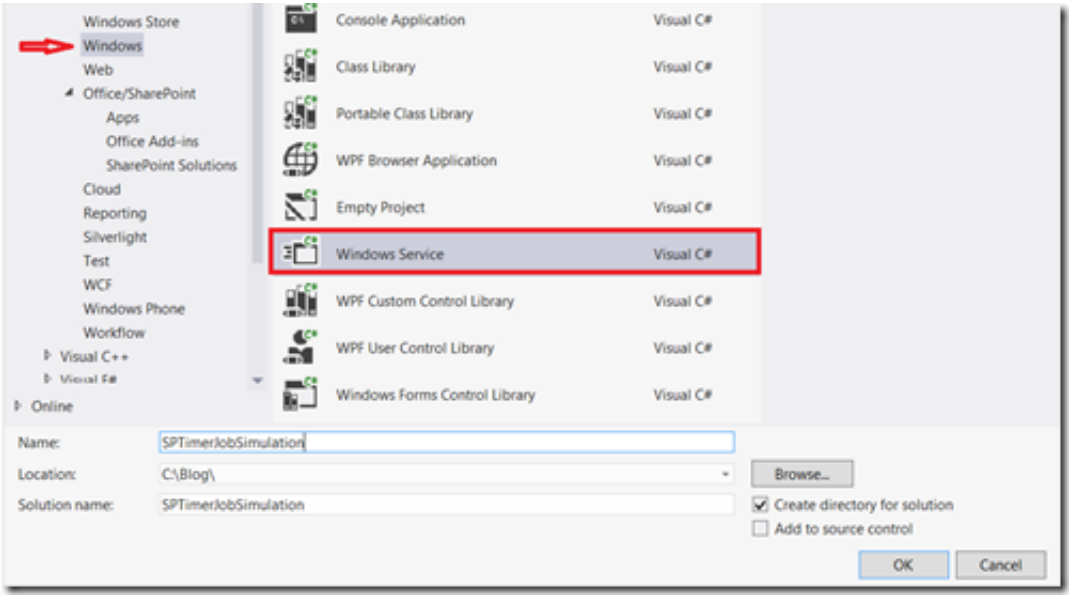
3. Please provide appropriate App permissions.

4. Now, once you are happy with the App functionality (mine just adds an item in a list). Verify the '**App Identifier**' with the '**App Display Name**' at **Site Settings -> Site app permissions** . Please note that another 'F5' Hit from VS 2012 would generate a new 'App Identifier' as VS will generate a new '**ClientId**'.

Step 2. Create a Windows Service that is scheduled to run [it utilizes the above code]

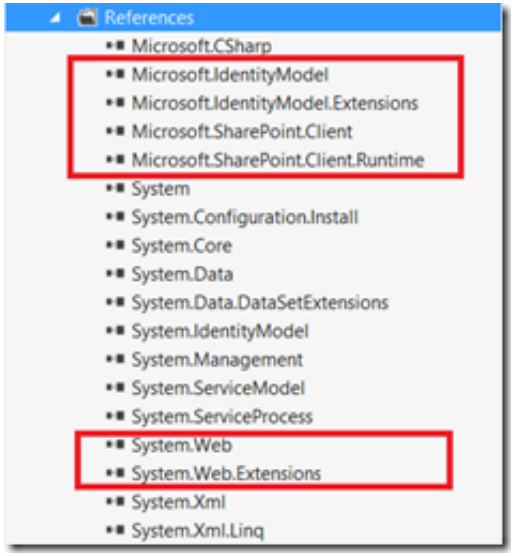
Check this How to: Create Windows Services, if you want to understand the details around creating a windows service that runs based on a schedule. Though I will mention it for everyone's interest.

1. Create a Windows Service by selecting **Windows Service** in the list of Visual C# project templates. I named it 'SPTimerJobSimulation'.



2. Add '**TokenHelper.cs**' class from the App project created above (Step 1). Then remove the namespace declaration from this class along with additional braces '{}' from the code.

3. Add the additional **references** (ref below). As this service needs to work with CSOM and other references needed for 'TokenHelper.cs' class.



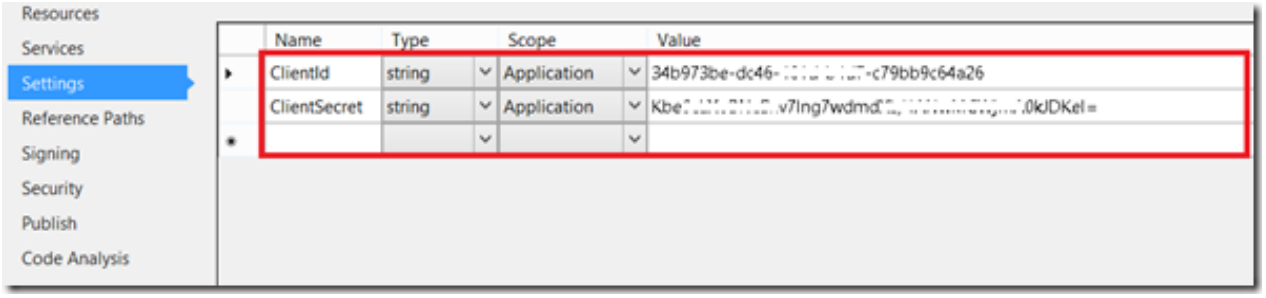
4. Open '**Service1.cs**' and switch to code view. I have pasted almost the entire code below. Essentially, I have moved the code from Default.aspx.cs. Just make sure to comment out the hard coded Uri to point to the relevant On Premise or SP-O site.

```
public partial class Service1 : ServiceBase
   {
      private static string SharePointPrincipal = "00000003-0000-0ff1-ce00-000000000000";
      static Timer SPtimer;

      public Service1()
      {
         InitializeComponent();
      }

      protected override void OnStart(string[] args)
      {
         SPtimer = new Timer();
         //schedule an interval of 2 minutes
         SPtimer.Interval = 1000 * 60 * 2;
         SPtimer.Elapsed += SPtimer_Elapsed;
         startSPtimer();
```

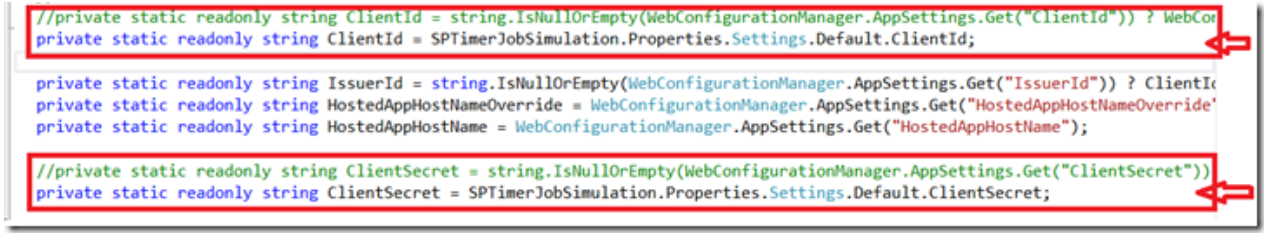
```
        }

        void SPtimer_Elapsed(object sender, ElapsedEventArgs e)
        {
            //Comment out either of the below based on On Premise server or Online
            Uri hostWeb = new Uri("http://tenant/sites/DevCenter");

            //Below line works with On line
            Uri hostWeb = new Uri("https://tenant.sharepoint.com/sites/Develop");

            string realm = TokenHelper.GetRealmFromTargetUrl(hostWeb);

            string appOnlyAccessToken = TokenHelper.GetAppOnlyAccessToken(SharePointPrincipal,
hostWeb.Authority, realm).AccessToken;

            using (ClientContext clientContext =
TokenHelper.GetClientContextWithAccessToken(hostWeb.ToString(), appOnlyAccessToken))
            {
                if (clientContext != null)
                {
                    var myList = clientContext.Web.Lists.GetByTitle("ShariqTest");
                    ListItemCreationInformation listItemCreate = new ListItemCreationInformation();
                    Microsoft.SharePoint.Client.ListItem newItem = myList.AddItem(listItemCreate);
                    newItem["Title"] = "Added from Timer Job";
                    newItem.Update();
                    clientContext.ExecuteQuery();
                }
            }
        }

        private static void startSPtimer()
        {
            SPtimer.Start();
        }

        protected override void OnStop()
        {
        }
    }
```

5. Next up, we need to add the Application settings (ref below). So go to Project settings and Select

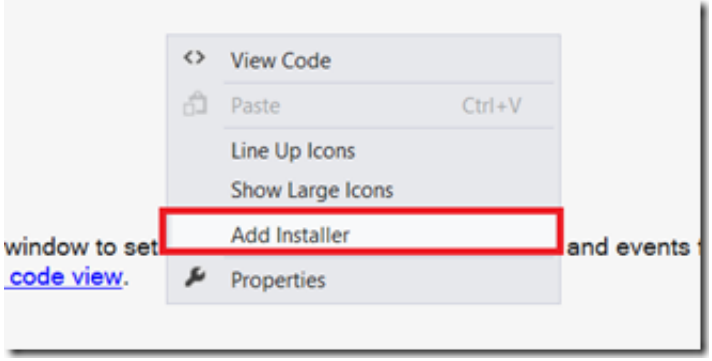
'**Settings**' to add the relevant <appSettings> from the Web.Config of your App project. Add '**ClientId**' & '**ClientSecret**' with relevant values (point '3', under Step 1 above). Also change scope to 'Application'.



Edit: Also you need to open your TokenHelper.cs file and comment out **private static readonly string ClientId** and replace it with yourAppsNamespace.Properties.Settings.Default.ClientId. Same goes for **private static readonly string ClientSecret,** replace it with yourAppsNamespace.Properties.Settings.Default.IssuerId.

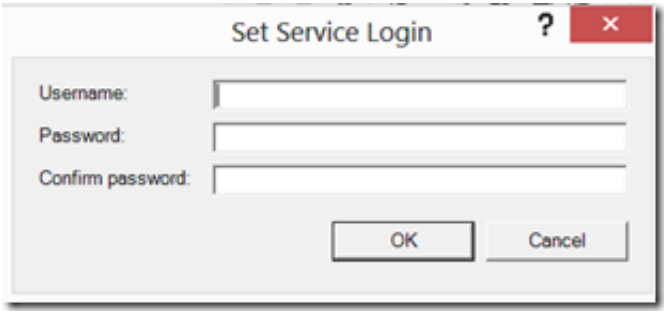


6. We are almost done. Now we need to Debug and Install this windows service, let's create the installers by opening **Service1.cs** file and with the designer in focus, right-click, Add Installer (ref below).



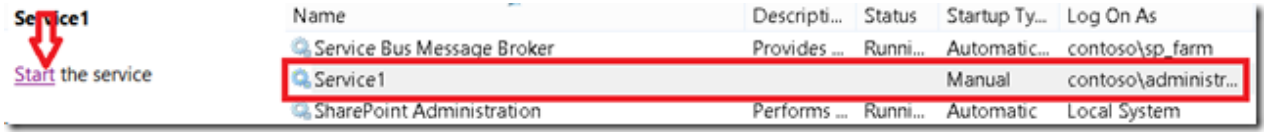
7. To install a Windows service. Open **Developer Command Prompt** as Administrator. Navigate to the folder that contains your project's output. Install the service using the 'installutil.exe' command (ref below).

8. It will prompt you for Service Login credentials. Provide appropriately.

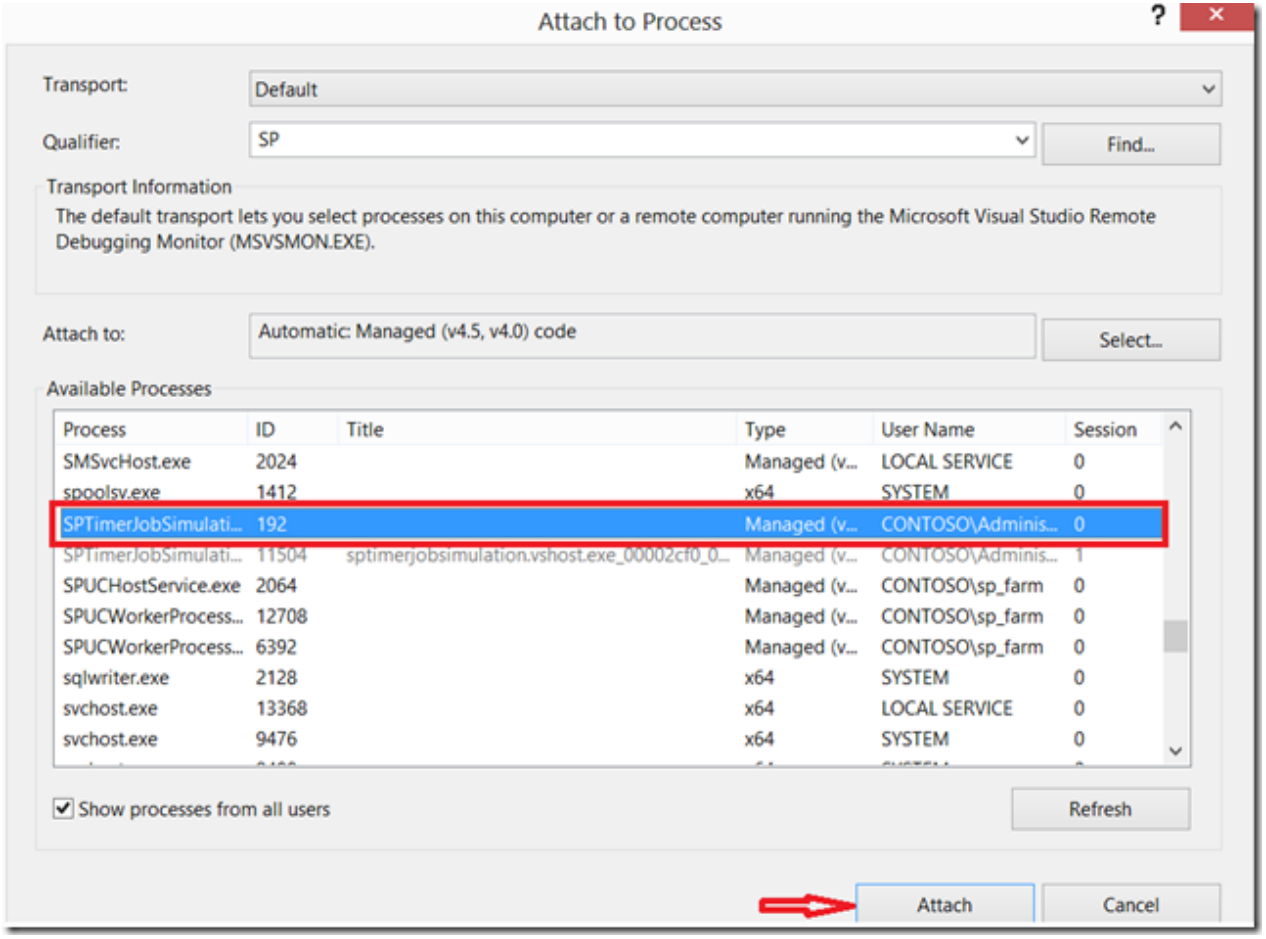


9. The Service would be installed (ref below).

10. Open '**Services**' and you should see your service listed. Go ahead and **Start** our service (ref below).



11. Now let's go and debug the service. Go to Debug –>Attach to Process and attach relevant process.



12. Now you should be able to debug your service. It should hit the breakpoint based on the schedule that you coded under OnStart method.

Now this Scheduled Service adds an item every 2 minutes to my SharePoint site :). But it could do anything (possible with CSOM) on either SharePoint On-Premise or Online based on your requirement. There might be an additional request around hosting this Service in **Azure** specially while you are working with SP-O. So a nice pattern could be:

- Host the workload processing logic inside a regular Azure worker role.
- Use Windows Azure Scheduler to queue work items in recurring scheduled fashion.
- Have the Azure worker role pick-up work items from queue and execute the workload. After the work is finished, delete the item from queue.

A couple of scenario which made me write a SharePoint timer job in the 2010 world was:

- Sending a reminder email based on a particular list value. This was used with a workflow.
- Updating SharePoint List data to have fresh data based on an external database change.

So, both these scenarios can be easily implemented with my Custom Timer Job simulator that doesn't use SSOM.

You can **download the complete sample** from here.

This is super powerful and I enjoyed it. Hope you like this too!

Your comments/suggestions are welcome, as always!

Cheers !

Shariq

Comments

Indrajit MSFT 11 Dec 2013 9:35 AM #
Great job! This is a great help to the community.

Vesa Juvonen 30 Dec 2013 10:39 PM #
Really nice example with detailed guidance. Thanks for sharing this to the community.

Fernando Toledo 13 Jan 2014 5:07 PM #
Hi.. I try use the example, but i get a error! :(

Token request failed.

The remote server returned an error: (400) Bad Request.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Net.WebException: The remote server returned an error: (400) Bad Request.

Shaan 2 Feb 2014 10:40 PM #

How to create custom timer job for an app containing list in sharepoint 2013 any example would be very helpful

Shariq-MSFT 11 Feb 2014 12:20 PM #

@Fernando - I missed one step, that you need to make enteries in your TokenHelper.cs file for ClientId and ClientSecret. Its added now under point 5 of Step 2 above. So you need to comment out private static readonly string ClientId and replace it with yourAppsNamespace.Properties.Settings.Default.ClientId. Same goes for private static readonly string ClientSecret, replace it with yourAppsNamespace.Properties.Settings.Default.IssuerId.

I have also uploaded the updated code. Thanks for pointing this out!!

Mark Stokes - 20 Feb 2014 1:59 AM #

Hi Shariq,

Thanks for sharing this, I am looking into a scheduled service to create SPOnline sites automatically from a On-Premises site request list and I think this approach could be the basis (albeit running in Azure as you explain at the end).

I am a little confused though over the use of the App? As far as I can tell the service you create creates a list item, as does the app. But when the Service runs every 2 minutes, how is the App involved?

I don't quite see the connection.

Thanks

Nick 4 Mar 2014 9:06 PM #

Hi Shariq!

Where to obtain a Client Secret?

Thanks

Nick

Shariq-MSFT 11 Mar 2014 2:32 PM #

@Nick - Visual Studio Autogenerates it for you. Or you can use the http://yourSharePointServerName/_layouts/15/appregnew.aspx URL to create a new one as mentioned at msdn.microsoft.com/.../jj687469(v=office.15).aspx

Shariq-MSFT 11 Mar 2014 2:33 PM #

@Mark Stokes - I use the Cloud App Model to demonstrate the ability of Apps/CSOM/App only policy. Then I take the same piece of code (used inside App) and use it in my Windows Service. It's independent.

Sudhir Rawat 9 Jun 2014 2:36 PM #

can we make a timer job which can access all the site collections of the farm?

Markus Ziegler 24 Jun 2014 5:55 AM #

Hello Shariq.

I'd like to thank you for this excellent guide!

The only problem i had during the implementation was the AllowAppOnlyPolicy option in the appmanifest. But now it works.

Thank you!

Vaibhav 27 Jun 2014 10:50 PM #

@Sudhir Sir: IMO, you can't. Client OM (Managed or JSOM) are limited in that way.

Shariq-MSFT 7 Jul 2014 10:47 AM #

@Sudhir Rawat - Yeah you should be able to just use the Tenant Admin creds

Shariq-MSFT 7 Jul 2014 10:48 AM #

@Markus Ziegler - AllowAppOnlyPolicy is the most important piece. Glad it worked!

Elena 17 Sep 2014 11:14 PM #

Hi Shariq, I'd like to access a list in the appWeb from the timer service. Is it possible to create a context to the appWeb?

Shariq-MSFT 12 Nov 2014 9:54 AM #

@Elena: sorry for the late reply.....you should be able to do that. I don't see a challenge here because app has complete access on the app. Are you facing any issues?

Kiran 12 Jan 2015 2:15 PM #

Hi Shariq

Please advice how to call a windows service from a sharepoint timer job?

Thanks

Sam 4 Mar 2015 2:46 AM #

An exception of type 'Microsoft.IdentityModel.SecurityTokenService.RequestFailedException' occurred in Microsoft.IdentityModel.Extensions.dll but was not handled in user code. I get the following error while running

from visual studio..whats wrong..